An Error-Minimizing Software Audit Technique

J. C. Holland and W. O. Paine
Quality Assurance—DSN and Mechanical Hardware Section

This article presents a method for systematizing a software code-design audit, using principles of set theory to delineate, with a minimum of effort and a maximum of error-detecting capability, the various individual discrepancies present in the software.

This analysis outlines the methodology of an auditing technique for evaluating computer software, which minimizes the effort involved, maximizes the information obtained from an audit, and minimizes auditing errors.

The process of uncovering discrepancies of whatever type between documentation of computer software systems and listings of software source programs is one of regarding labels and the sections of logic[1] so labeled (both in the assembly listing and in the documentation) as members of sets. Clearly, labels either agree (aside from typographical errors) or they do not. The situation is less clear when comparing sections of logic, which may be further subdivided, when necessary, into smaller sections—those which agree between the documentation and the source program, and those which do not. We may repeatedly subdivide sections of logic, as required, until we reach the point at which individual executable statements are compared. The "sections of logic," then, are simply collections of consecutive executable statements, separated from one another by whatever condition is encountered which creates a natural boundary. This may be the beginning or end statements of a subroutine or control program, a label, or an encounter with statements in the logic of the documentation or the source program which have no counterpart in the other.

These collections of consecutive executable statements (of whatever extent) and labels are thought of as the elements of sets. By the process of finding the intersections of these sets and their complements we may then establish various categories of discrepancies. This approach of uncovering discrepancies has three very powerful advantages over the examination of each labeled section of logic in the assembled program or in the documentation on an individual basis:

(1) By treating whole classes of labels or sections of logic having the same characteristics as a single

[1] Here, and elsewhere in this article, the phrase "section of logic" is taken to mean any set of consecutive executable statements regardless of their extent.

entity, we avoid overlooking some of the (otherwise) harder-to-identify discrepancies. We also avoid describing them incorrectly because of failure to notice some of of their characteristics.

(2) It is inherent in the process described below that it proceeds much more rapidly than an individual item-by-item examination and description.

(3) The use of set-theoretic processes has extremely powerful self-inductive properties. We get more out of the process than we put in. The completion of each step in the described procedure immediately suggests ways to use information already obtained in other ways.

One may regard the universe of elements being treated by the logical model as consisting of all labels and all sections of logic, both in the documentation and in the assembled program. These four categories, or sets, intersect in a manner shown in Fig. 1, the various subsets of which—intersections and their complements—are labeled with Roman numerals and subsequently described.

I. *Complete correspondence*—A logic section is found both in the documentation and the source program listing and is labeled the same in both. The logic completely agrees between the two. This corresponds to ABCD of Figure 1.

II. *Dummy section of labeled assembly-listing logic*—The labels in the source listing and the documentation agree and the documentation logic is given, but the source listing has plugged-up, or non-existent, logic. Note that this condition corresponds to $ABC\bar{D}$ (A and B and C and NOT-D) and immediately reveals its characteristic—unimplemented logic—by being inside circles (sets, that is) A, B, and C, but outside D.

III. *Undocumented label*—The logic exists both in the documentation and source listing, but is labeled only in the source listing. This corresponds logically to $\bar{A}BCD$.

IV. *Undocumented logic section*—This label exists both in the source listing and in the documentation (by reason of being the title of a program block within a subroutine or control program but not the title of a separate flow chart or section of flow chart) and corresponds to a section of logic in the source listing. This is, of course, $A\bar{B}CD$.

V. *Unlabeled or incorrectly labeled section of code in the source listing*—The logic is in the documentation, where it is labeled, and in the source listing, where it either is an unlabeled section of code embedded in another program section, or has a different label. This corresponds to $AB\bar{C}D$.

VI. *Documented, labeled code not in the source listing at all*— Unimplemented code not represented by a labeled program stub or dummy. This is $AB\bar{C}\bar{D}$.

VII. *Documented logic, unlabeled, in the source listing as a label only, no logic*—This is a strange case, but it could happen. The nature of the logic (and the names used) in the documentation could be so unique that the dummy label in the source program could correspond *only* to that section of logic in some undefined way. Comments in both places could also correspond. This is one example of how the set-theoretic approach gives us more than we asked for, and is identified as $\bar{A}BC\bar{D}$.

VIII. *Source listing logic; labeled, not documented*—Self-explanatory. Corresponds to $\bar{A}\bar{B}CD$.

IX. *Documented label (no documented logic) corresponds to an unlabeled section of source listing logic*—This, by the way, is the complement of case VII. Here the nature of the operations being performed in the source program logic is so unique (or accompanied by comments) as to identify them with the label in the documentation (in a logic-block label within another subroutine, say), even though that logic is never documented. This is $A\bar{B}\bar{C}D$.

X. *Documented label only, no documented logic and no such label or logic in the source program*—This is the case where a single logic block in a documented subroutine or control program has a label which is found nowhere else—neither in the documentation nor the source program. This is identified logically as $A\bar{B}\bar{C}\bar{D}$.

XI. *Documented section of logic, unlabeled, unimplemented*—Here is the case where (as, for example, in a control program originated in the lower memory) no label is attached, and the program does not exist in the source Program. This is $\bar{A}B\bar{C}\bar{D}$.

XII. *Dummy source listing label*—No documentation logic or label and no corresponding section of logic as such in the source program. Corresponds to $\bar{A}\bar{B}C\bar{D}$.

XIII. *Unlabeled source listing section of logic, undocumented*—Could indicate revision of program since documentation was last updated. This is $\bar{A}\bar{B}\bar{C}D$.

Considering that the above classification rests upon the basis of set-intersections and their complements, one should proceed to classify the various elements (labels and sections of logic) by beginning with the most inclusive, as well as most easily managed—by reason of being alphabetically ordered—set, the Autoflow[2] label index, and the set of documentation flow charts. Matching and nonmatching labels will then constitute separate categories which may further be subdivided by the identification of label index entries as undocumented subroutines, control programs or unidentified sections of implemented logic, and the matching of flow chart logic in the documentation with some sections of labeled or unlabeled logic in the source program.

Thus the repeated subdivision of the largest categories of labels and sections of logic into their various subsets produces the various categories of discrepancies without redundancy and with minimum effort.

The following is a complete description of the process of obtaining an audit by this method:

(1) Using the Autoflow flow chart set and its label index, mark each label in the index which corresponds to the name of a subroutine or (apparently, judging by comments or (later) by documentation flow charts) control programs.

(2) Collate the documentation flow charts with the Autoflow label index, putting the names which match on a given list (call it List 1) if the label index entry is marked (indicating a subroutine or control program) and on List 2 if the label index entry is unmarked. Place the names of unmatched documentation flow charts on List 3. Additionally, we have the unmatched labels of the Autoflow label index which are also unmarked; call them List 4. These lists are illustrated in Figure 2.

(3) Compare the logic of the source programs on List 1 with the logic of the corresponding documentation flow chart. One of three conditions should be observed:

   (a) Complete agreement between the two. *This is Category I* of the preceding classification.

   (b) Agreement between the two except for isolated discrepancies (individual steps omitted from one or the other, incorrect order of steps, lack of YES/NO labeling on decision steps, etc.). These will be *reported on the Discrepancy List of the audit report.*

   (c) Entire sections of logic (labeled or unlabeled) existing in one of the two, but not the other. *Mark the extent of this logic block* on the Autoflow chart or documentation chart, wherever it occurs. Put its beginning label (if any) or location on a separate list as noted below.

   We have now separated List 1 into four sublists, shown in Figure 3.

(4) Compare the logic of the source programs and subroutines on List 2 with the documentation flow charts having the same labels. This will have the effect of separating this list into sublists in a manner similar to that used to partition List 1. This breakdown is shown in Figure 4.

(5) Compare the extraneous blocks of logic whose beginning addresses are found in Lists 1C and 2C with the unmatched documentation flow charts found in List 3. The various outcomes are shown in Figure 5.

(6) Using the Autoflow label index and concordance, place those labels not already marked on List 6A if they are entirely unreferenced (in the concordance), on List 6B if they are only internally referenced (according to the concordance). Additionally, an examination of the Autoflow charts will reveal a certain number of initial points in lines of flow that are unlabeled and reveal no apparent method of access. Since these may conceivably be reached by an indexed branching instruction, this condition should be checked for. If it cannot be determined that this is the case and, labeled or unlabeled, there seems to be no way to reach these sections of logic, their initial locations should be entered on List 6C. The results are shown in Figure 6.

(7) Compare the logic blocks found only in the documentation flow charts (Lists 1D and 2D) with the unmatched sections of source program logic from Lists 5D and 6A, B and C. The various outcomes are presented in Figure 7.

(8) Finally, make lists of documentation labels which are unaccompanied by corresponding logic (e.g., being representative of some undefined program stub) and source program labels unaccompanied by logic (perhaps representing dummy sections of unimplemented source program code). Compare the documentation labels with unidentified sections of source program logic from Lists 6A, B and C, in the sense that unique identifiers may appear in the comments of both or in the names of operands used in the source program logic. Compare the source program labels from above with the extraneous logic blocks from Lists 7A and B in the sense that the two

[2] Registered trademark of Applied Data Research, Inc. (Ref. 1).

may use the same unique identifiers or correspond with respect to their comments. The outcomes of these comparisons are shown in Figure 8.

From all of the above classifications of errors, the software discrepancy audit report may be written, treating entire classes rather than individual errors, the exception being the descrepancy list, in which the differences between individual blocks or labels in the source program and documentation are discussed in detail.

As an example of the foregoing analysis, we consider the two flow charts of Figures 9 and 11, headed by the entry points labeled "RQSTCK" and "STOPCK," respectively:

(1) In step 1 of the previously described procedure no such labels as RQSTCK or STOPCK were encountered in the Autoflow label index nor, of course, in the Autoflow charts themselves. A label, CONTRL, which matched a subroutine, was found and marked.

(2) In step 2 of the procedure it was noted that the documentation flow charts for RQSTCK and STOPCK were unmatched by corresponding labels in the Autoflow label index, and therefore these two flow chart names were placed on List 3.

   The documentation flow chart for CONTRL, being matched by an Autoflow chart and marked label in the label index, was placed on List 1.

(3) In step 3 of the above-described procedure, the logic of the documentation flow chart CONTRL was compared block-by-block with the logic of the Autoflow chart headed by the same label—CONTRL. The points at which the logic of the documentation flow chart for CONTRL failed to agree with that of the Autoflow chart were marked for future reference. Sizeable sections of the Autoflow logic were found to have no counterpart in the documentation flow chart. Their extent was marked, as in step 3c. There was no beginning label.

(4) Step 4 of the procedure was not involved in this case.

(5) In step 5 of the procedure, a match was found between the logic of two of the flow charts on List 3—STOPCK and RQSTCK—and parts of the unmatched logic of the CONTRL Autoflow chart. The match was in both cases less than perfect. At this point the logic blocks which did match between CONTRL and either STOPCK or RQSTCK were marked, and a line-by-line examination of the source program code and the unmatched logic (or missing logic, as the case may be) of the documentation was undertaken to pinpoint the differences. The documentation flow charts were corrected accordingly, as shown in Figures 10 and 12.

   In both cases (STOPCK and RQSTCK) the differences in logic between the documentation flow charts and source program code were noted in the discrepancy list, and the fact that these so-called "subroutines" were in reality sections of unlabeled code embedded in another program was indicated by their presence on the List 5B, which is reported by classification in the audit. The annotated source program code and Autoflow charts are given in Figures 13 and 14, respectively.

Reference

1. *Map Autoflow II Assembly Series Reference Manual,* Applied Data Research, Inc. (Copyright January 1974 by Applied Data Research, Inc.).

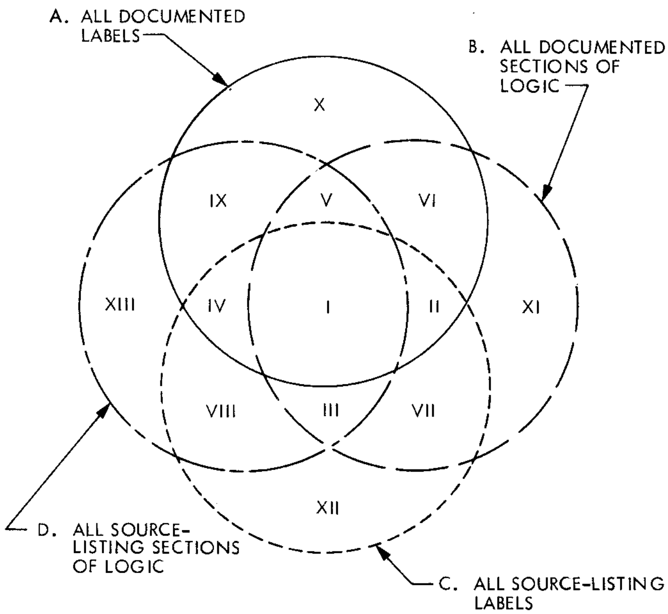


Fig. 1. A Venn Diagram of the sets of labels and logic sections, and their intersections

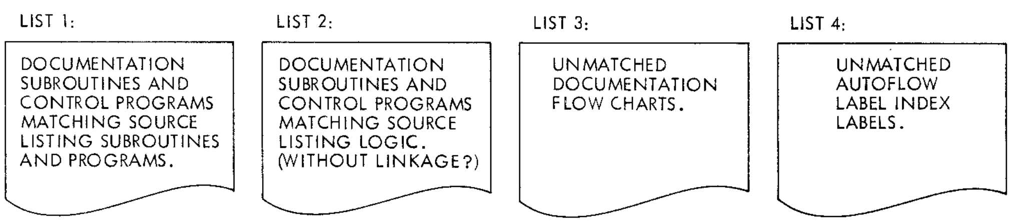


Fig. 2. An initial breakdown of matching and non-matching labels

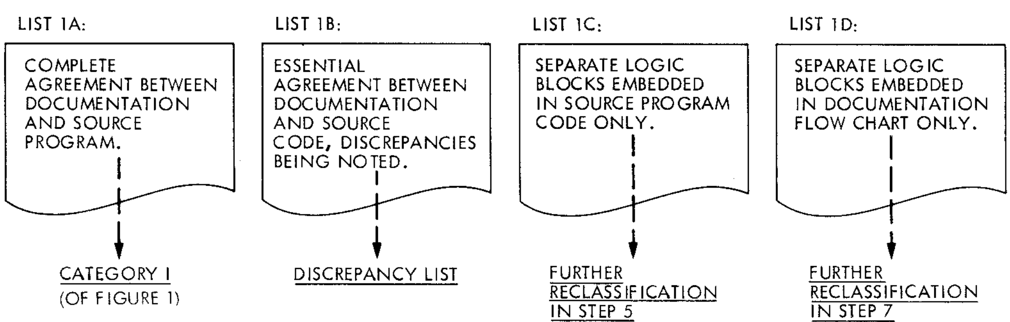


Fig. 3. A breakdown of matching and non-matching sections of logic between similarly labeled program sections

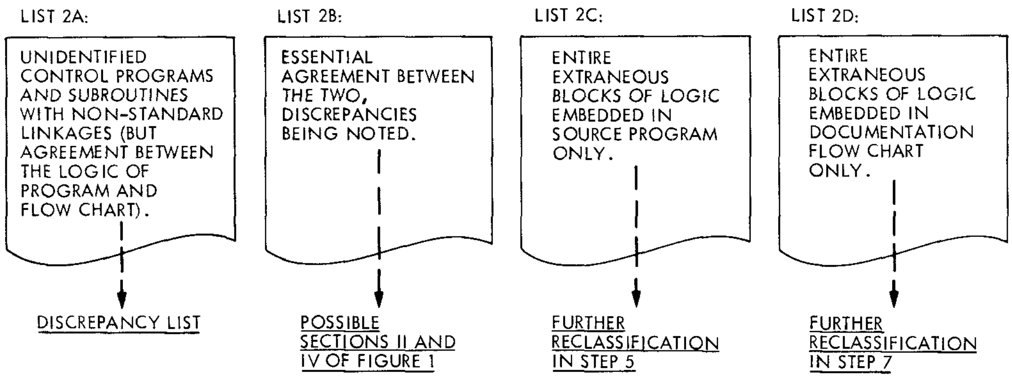


Fig. 4. A breakdown of matching and non-matching sections of logic between dissimilarly labeled program sections

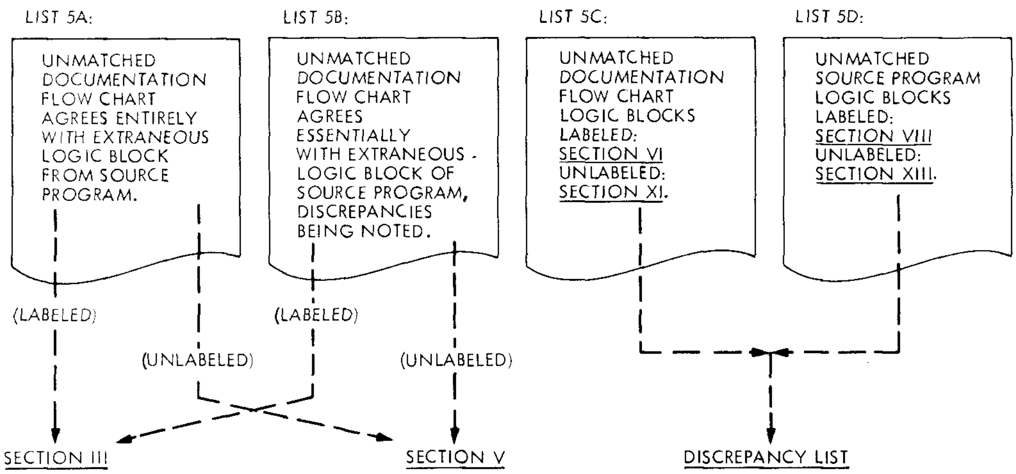


Fig. 5. A breakdown of matching and non-matching sections of logic between unmatched flowcharts and extraneous source-program logic

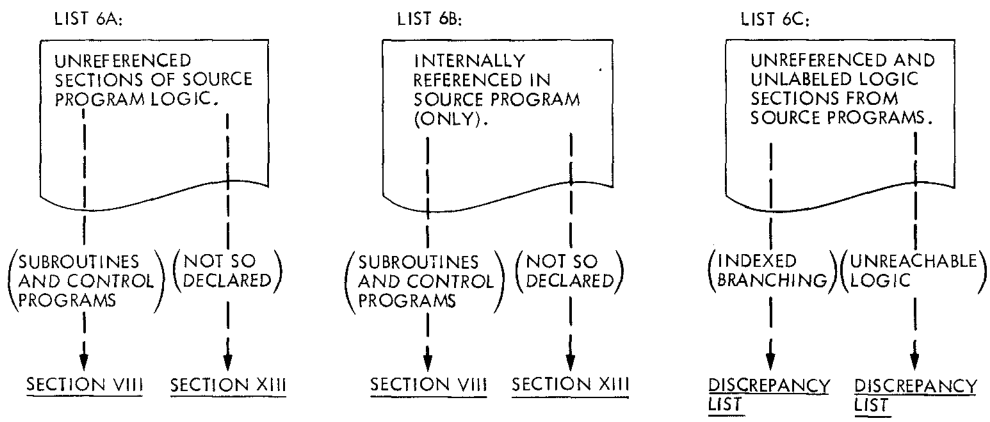


Fig. 6. Completely unreferenced sections of flow chart or source program logic

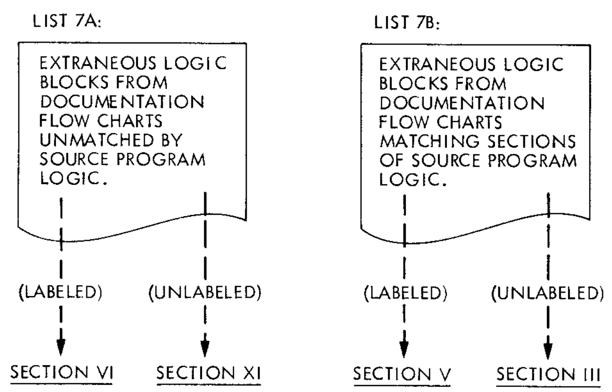


Fig. 7. Collation of unreferenced logic sections from flow charts and source program

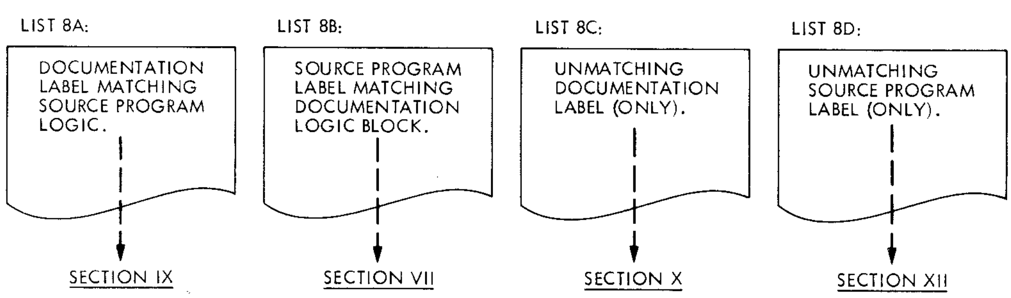


Fig. 8. A breakdown of matching label/logic block pairs and unmatching labels

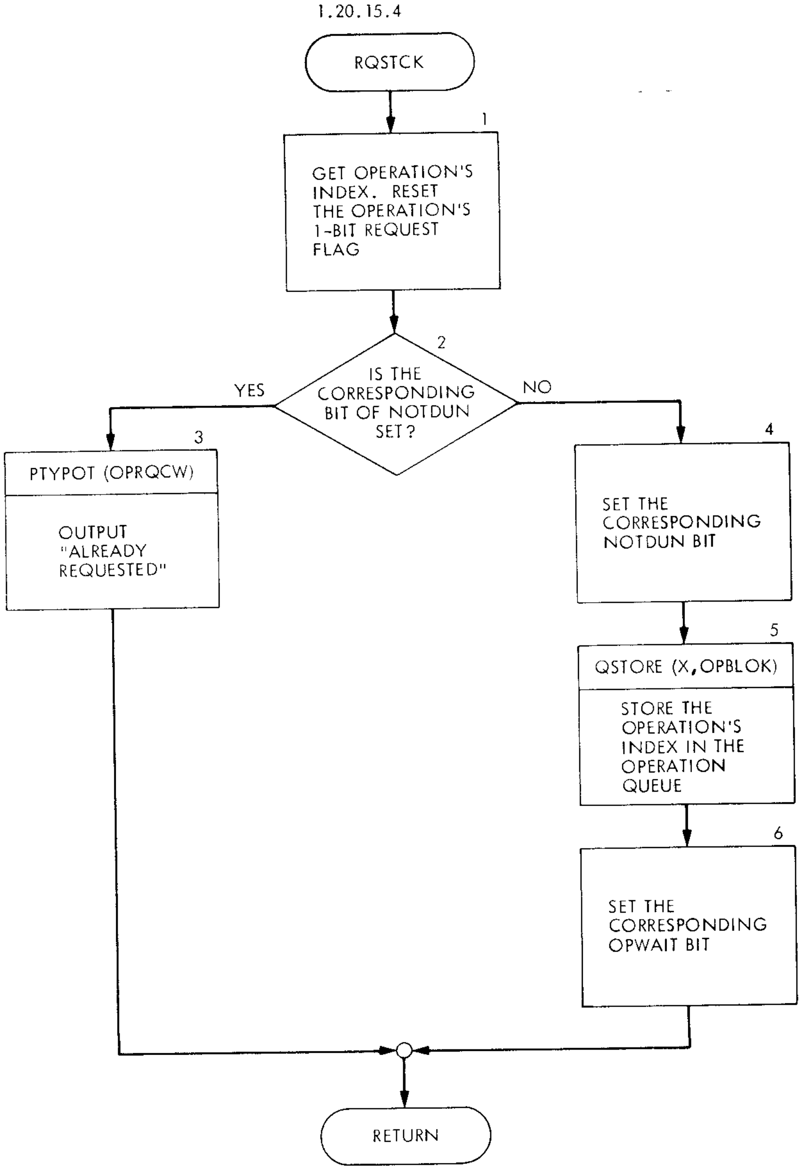


Fig. 9. Original flow chart from documentation

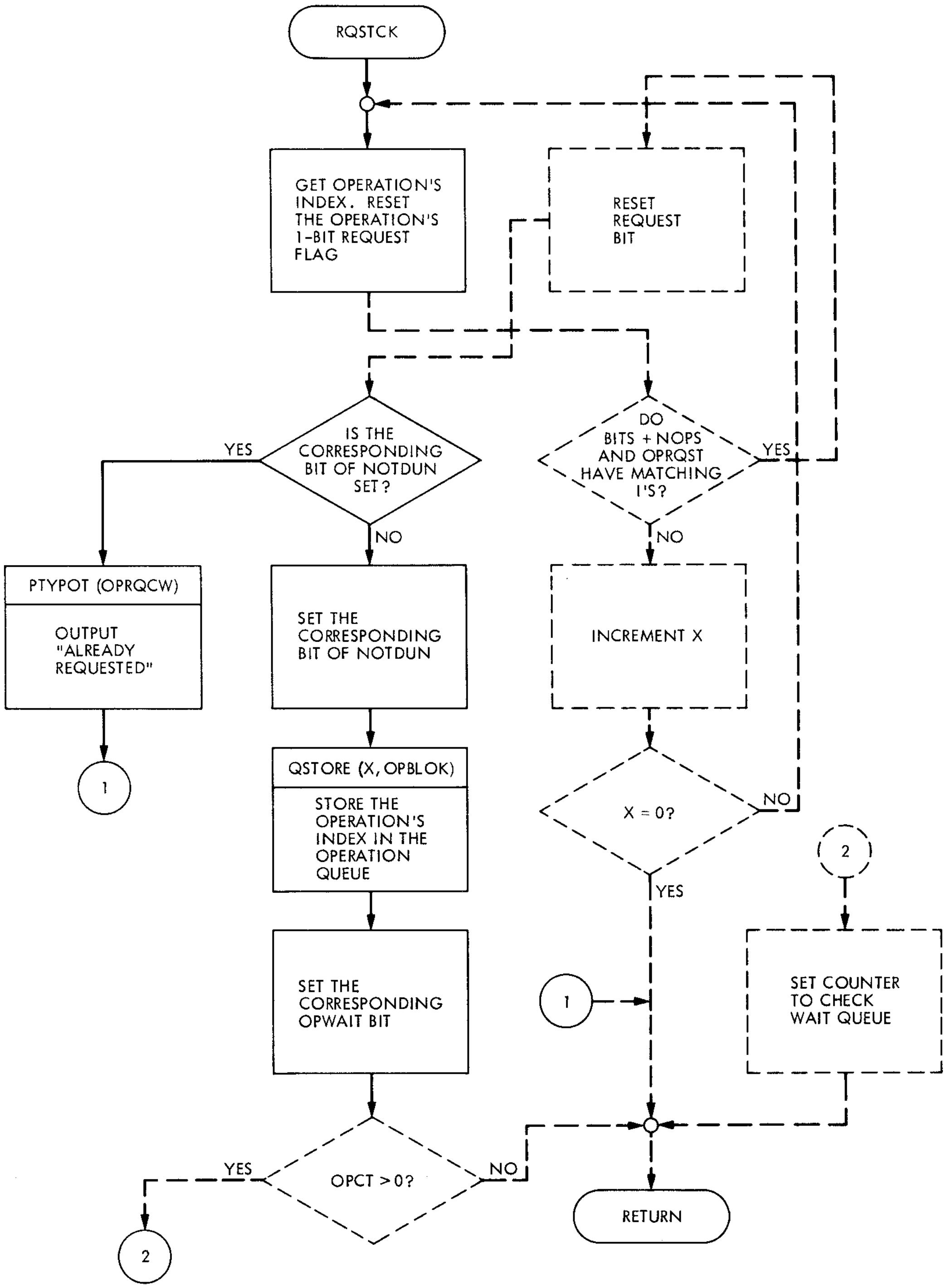


Fig. 10. Corrected flow chart agreeing with source program

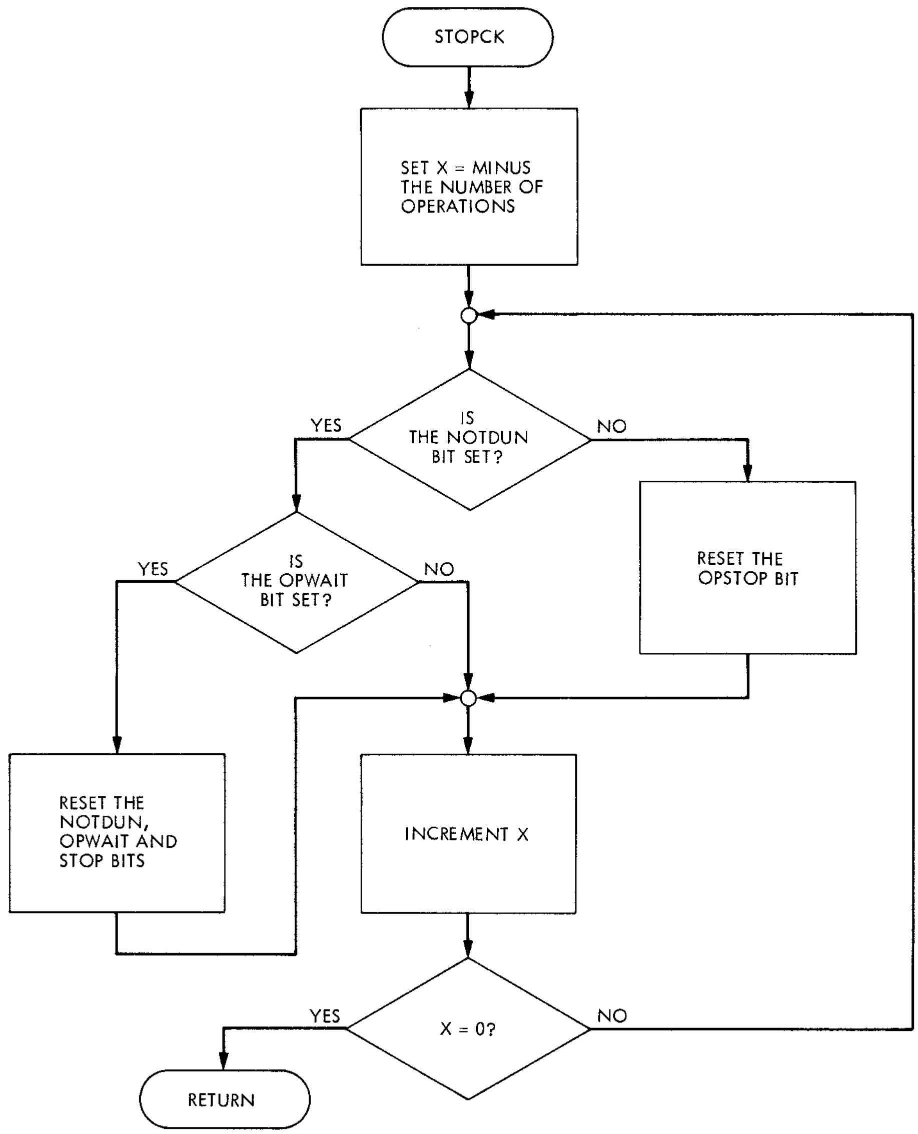


Fig. 11. Original flow chart from documentation

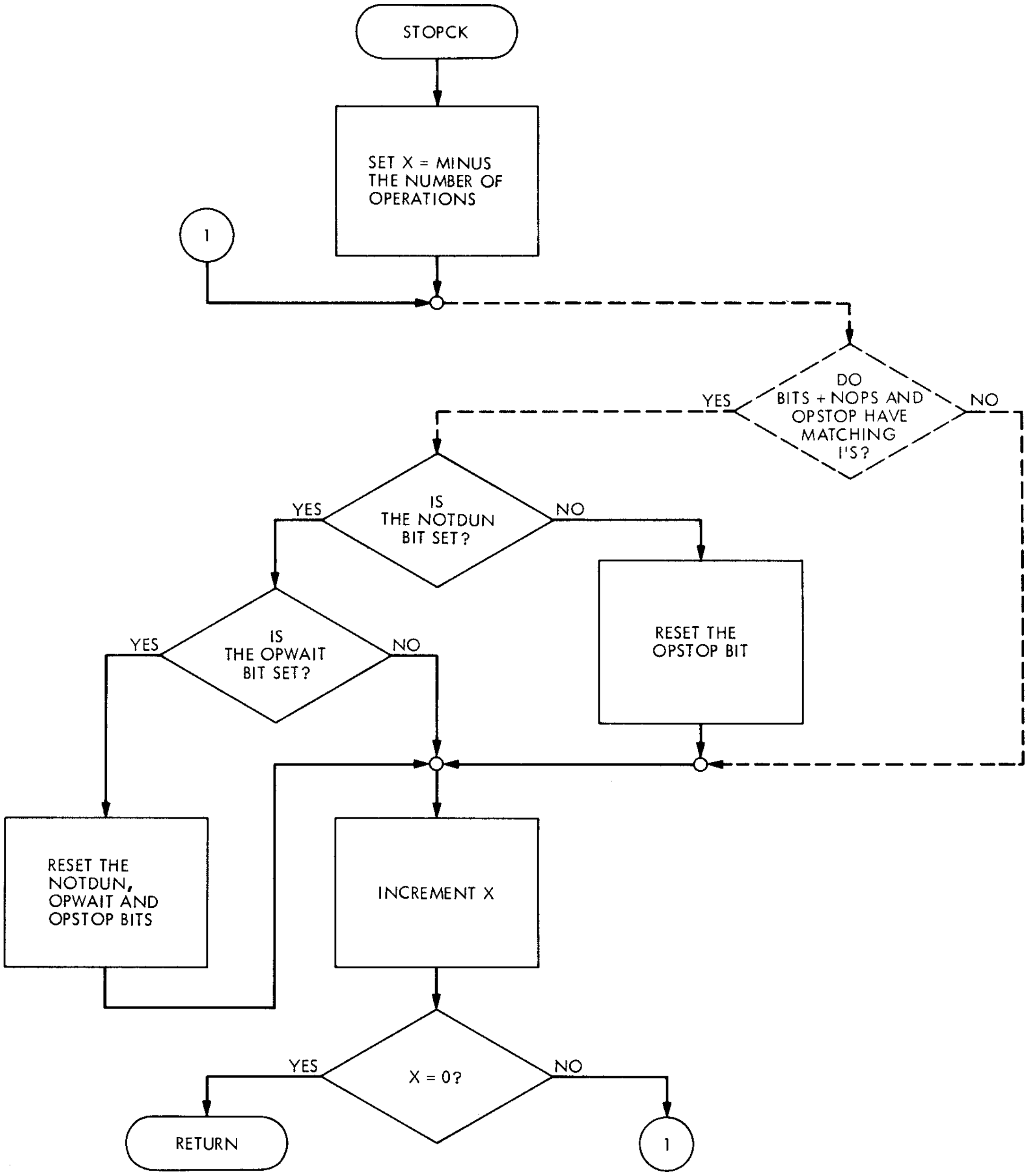


Fig. 12. Corrected flow chart agreeing with source program

```
         BRR     CK2BLK                    RETURN                            MDG01462
*                                                                            MDG01464
*        THIS ROUTINE STORES A HSD OUTPUT BLOCK AND CHECKS THE STOP BIT      MDG01466
*         (B) HOLDS THE OPERATION'S MASK                                     MDG01468
STRSTP   PZE     .                                                           MDG01470
         SKB     OPSTOP   .                                                  MDG01472
         BRU     $+2   .                                                     MDG01474
         BRU     $+4   .                                                     MDG01476
         BRM     RELES1                                                      MDG01478
         BRM     LSTSEQ   .                LAST BLOCK IN THE FILE            MDG01480
         BRU     $+2   .                                                     MDG01482
         MIN     STRSTP   .                                                  MDG01484
         BRM     STROUT   .                                                  MDG01486
         BRR     STRSTP   .                                                  MDG01488
*********************************************************************MDG01490
*        CONTROL PACKAGE                                                     MDG01492
*                                                                            MDG01494
*                                                                            MDG01496
CONTRL   PZE     .                         CONTROL ROUTINE                   MDG01498
         LDA     OPSTOP   .                                                  MDG01500
         SKG     =0   .                    IS A STOP REQUEST SET?            MDG01502
         BRU     CNSTRT   .                NO, CHECK FOR A START REQUE       MDG01504
         LDX     =-NOPS   .                                                  MDG01506
         LDA     BITS+NOPS,2   .                                             MDG01508
         SKA     OPSTOP   .                IS A BIT SET?                     MDG01510
         BRU     $+3   .                                                     MDG01512
STPNXT   BRX     $-3   .                   NO, TRY NEXT BIT                  MDG01514
         BRU     CNSTRT   .                                                  MDG01516
         SKA     NOTDUN   .                YES, IS THE NOT DONE FLAG SET?    MDG01518
         BRU     $+4   .                   YES                               MDG01520
         EOR     OPSTOP   .                NO,  RESET THE OPSTOP BIT         MDG01522
         STA     OPSTOP   .                                                  MDG01524
         BRU     STPNXT   .                                                  MDG01526
         SKA     OPWAIT   .                IS THE OPERATION IN A QUEUE       MDG01528   STOPCK
         BRU     $+2   .                                                     MDG01530
         BRU     STPNXT   .                NO                                MDG01532
         EOR     OPWAIT   .                YES, RESET THE BITS               MDG01534
         STA     OPWAIT   .                                                  MDG01536
         LDA     BITS+NOPS,2   .                                             MDG01538
         EOR     NOTDUN   .                                                  MDG01540
         STA     NOTDUN   .                                                  MDG01542
         LDA     BITS+NOPS,2   .                                             MDG01544
         EOR     OPSTOP   .                                                  MDG01546
         STA     OPSTOP   .                                                  MDG01548
         BRU     STPNXT   .                                                  MDG01550
CNSTRT   LDA     OPRQST   .                                                  MDG01552
         SKG     =0   .                    IS A REQUEST BIT SET?             MDG01554
         BRU     CKWAIT   .                NO, CHECK FOR ANY WAITING OPERAN  MDG01556
         LDX     =-NOPS   .                                                  MDG01558
         LDA     BITS+NOPS,2   .                                             MDG01560
         SKA     OPRQST   .                                                  MDG01562
         BRU     $+3   .                                                     MDG01564
         BRX     $-3   .                   TRY NEXT BIT                      MDG01566   RQSTCK
         BRR     CONTRL   .                DONE                              MDG01568
         EOR     OPRQST   .                RESET REQUEST BIT                 MDG01570
         STA     OPRQST   .                                                  MDG01572
         LDA     BITS+NOPS,2   .                                             MDG01574
         SKA     NOTDUN   .                IS THE NOT DONE BIT SET?          MDG01576
         BRU     $+2   .                                                     MDG01578
         BRU     $+6   .                                                     MDG01580
         LDB     OPRQCW   .                YES, OUTPUT 'ALREADY REQUES'      MDG01582
```

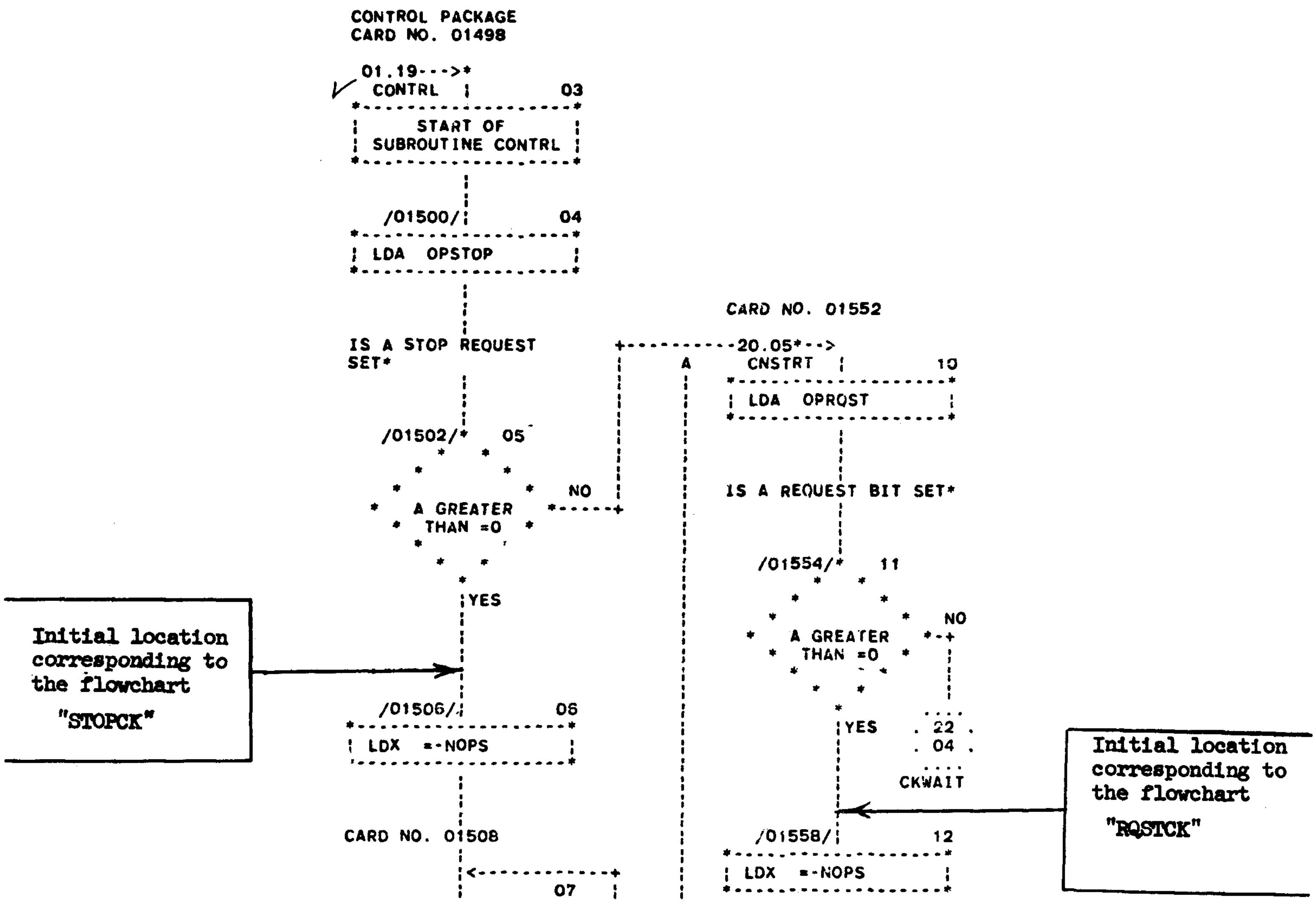
Fig. 13. Source program coding showing extent of unlabeled and improperly linked "subroutines" RQSTCK and STOPCK

```
        BRM     PTYPOT   .                                            MDG01584
        LDB     RETNCW   .        CARRAIGE RETURN                     MDG01586
        BRM     PTYPOT   .                                            MDG01588
        BRU     CKWAIT   .       CHECK FOR ANY WAITING OPERATIONS     MDG01590
        MRG     NOTDUN   .                                            MDG01592
        STA     NOTDUN   .       NO, SET IT                           MDG01594
        STX     OPIDX   .                                             MDG01596
        CXB                                                           MDG01598
        LDX     =OPBLOK                                               MDG01600
        BRM     QSTORE   .       STORE IN OPERATIONS QUEUE            MDG01602
        LDX     OPIDX   .                                             MDG01604
        LDA     BITS+NOPS,2   .                                       MDG01606
        MRG     OPWAIT   .                                            MDG01608
        STA     OPWAIT           SET WAIT BIT                         MDG01610
.CKWAIT LDA     OPCT                                                  MDG01612
        SKG     =0   .                                                MDG01614
        BRR     CONTRL   .                                            MDG01616
        SUB     =1   .                                                MDG01618
        STA     OPCNTR   .       SET COUNTER TO CHECK WAIT QUEUE      MDG01620
        LDA     MASKFG   .                                            MDG01622
        STA     OPBUSY   .       QUEUE HOLDS INDEX OF WAITINPERATIONMDG01624
.NXTOP  LDX     =OPBLOK   .                                           MDG01626
        BRM     QGET   .         GET INDEX OF OPERATION IN THE QE     MDG01628
        CBX .                                                         MDG01630
        LDA     BITS+NOPS,2   .  HAS THE OP BEEN CANCLED?             MDG01632
        SKA    *NOTDUN                                                MDG01634
        BRU     $+2              NO                                   MDG01636
        BRU     RESTOR+6   .     YES, DONT RESTORE IT                 MDG01638   RQSTCK
        LDA     OPBUSY   .                                            MDG01640
        SKA     BITS+NOPS,2   .  DOES THIS OPERATION CONFLICITH       MDG01642
        BRU     RESTOR   .       YES, PUT IT BACK IN THE QUEUE  OTHEMDG01644
        LDA     OPRTNS+NOPS,2   .  NO, INITIALIZE THE OPERATION       MDG01646
        STA*    ENTRYS+NOPS,2   .                                     MDG01648
        LDA     BITS+NOPS,2   .  RESET THE WAIT FLAG                  MDG01650
        EOR     OPWAIT   .                                            MDG01652
        STA     OPWAIT   .                                            MDG01654
        LDA     MASKS+NOPS,2   .                                      MDG01656
        MRG     MASKFG           MASK OUT CONFLICTION OPERATIONS      MDG01658
        STA     MASKFG                                                MDG01660
        BRU     $+4   .                                               MDG01662
.RESTOR LDX     =OPBLOK   .                                           MDG01664
        BRM     QSTORE   .       RESTORE THE OPERATION                MDG01666
        CBX .                                                         MDG01668
        LDA     MASKS+NOPS,2                                          MDG01670
        MRG     OPBUSY           MASK ANY OPERATIONS CONFLICG WITH    MDG01672
        STA     OPBUSY   .         THIS                               MDG01674
        SKR     OPCNTR   .       ANY MORE?                            MDG01676
        BRU     NXTOP   .        YES                                  MDG01678
        LDX     OPIDX   .        NO, CHECK TO SEE IF REQUESTED        MDG01680
        LDA     BITS+NOPS,2        OP. WAS STARTED                    MDG01682
        SKA     OPWAIT   .       WAS, OP STARTED?                     MDG01684
        BRU     $+2   .          NO                                   MDG01686
        BRR     CONTRL   .       YES, RETURN                          MDG01688
        CLA .                                                         MDG01690
        STA     OPIDX   .        RESET THE OPERATION'S INDEX          MDG01692
        LDB     DELACW   .       OUTPUT DELAY MESSAGE                 MDG01694
        BRM     PTYPOT   .                                            MDG01696
        LDB     RETNCW   .                                            MDG01698
        BRM     PTYPOT   .                                            MDG01700
        BRR     CONTRL   .                                            MDG01702
*          OPERATION CONTROL DATA                                     MDG01704
```

Fig. 13 (contd)

01/07/76 AUTOFLOW CHART SET - -(Q.A.)- DIS/HSD DOI-5468-SP 1/6/76 PAGE 20

CHART TITLE - DIS/HSD DOI-5468-SP 1/6/76



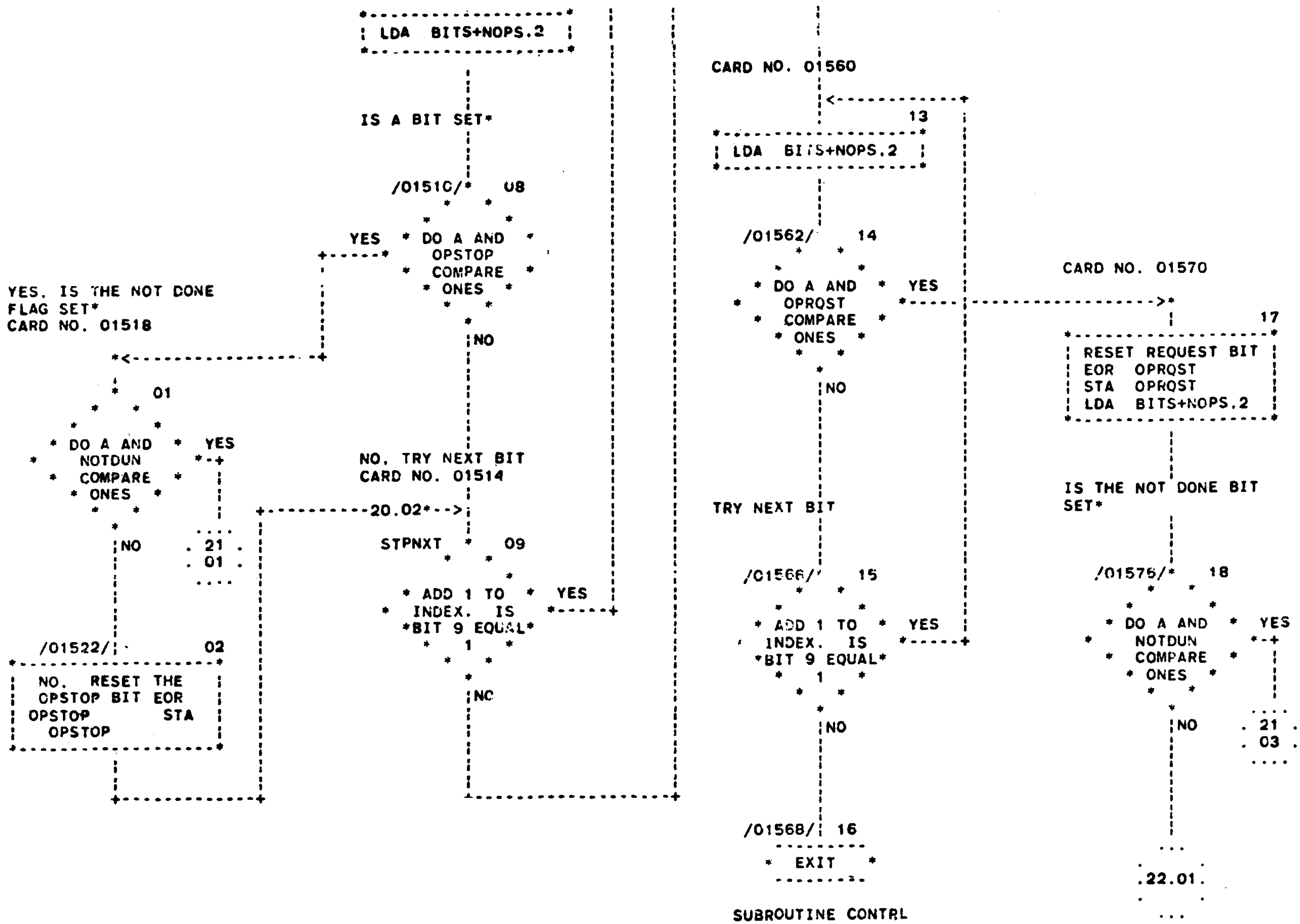


Fig. 14. Autoflow charts for the subroutines CONTRL, STOPCK, and RQSTCK

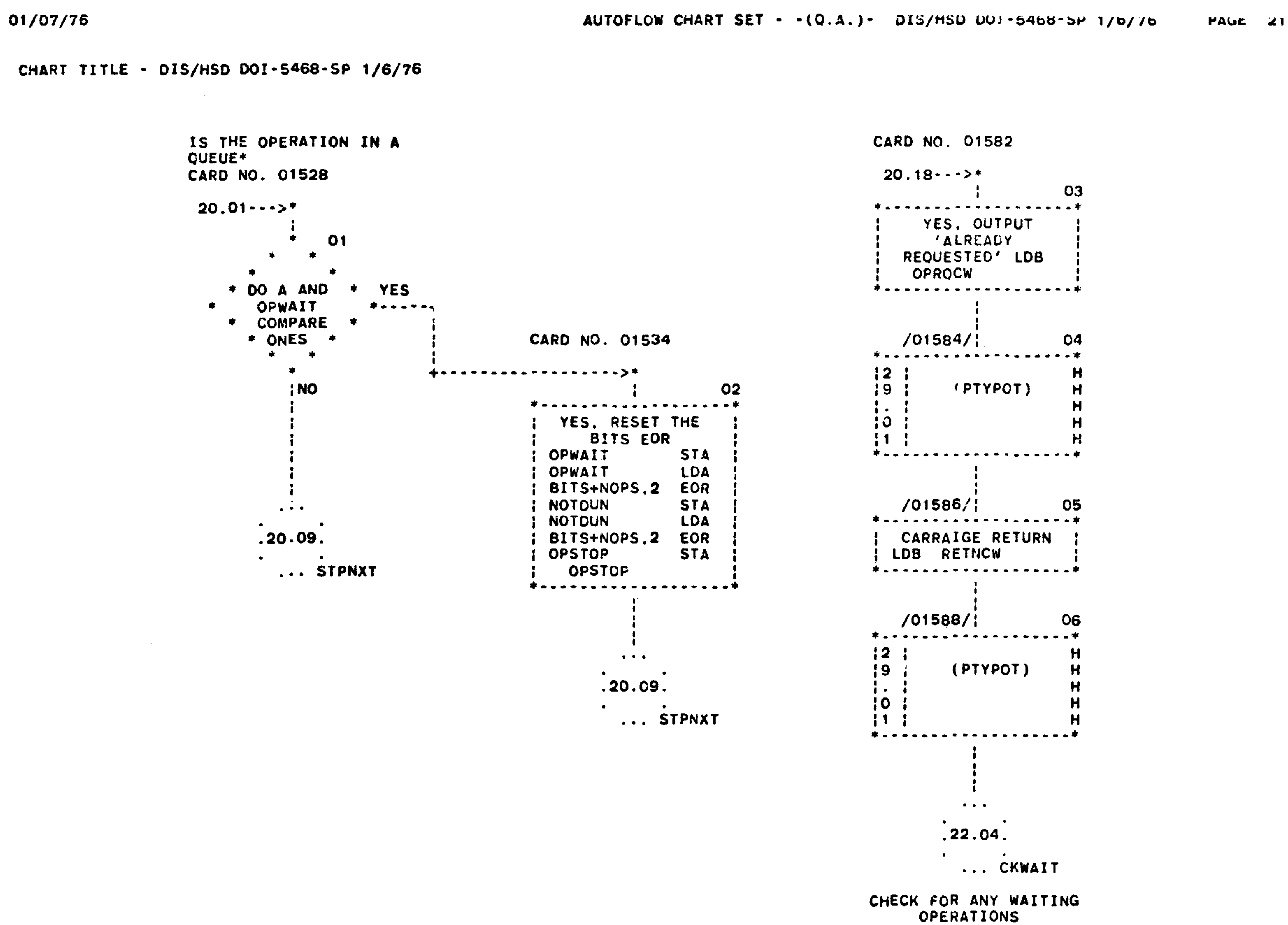

01/07/76
AUTOFLOW CHART SET - -(Q.A.)- DIS/HSD DOI-5468-SP 1/6/76
PAGE 21
CHART TITLE - DIS/HSD DOI-5468-SP 1/6/76
IS THE OPERATION IN A QUEUE*
CARD NO. 01528
20.01--->*
01
DO A AND OPWAIT COMPARE ONES
YES
NO
.20.09.
... STPNXT
CARD NO. 01534
02
YES, RESET THE BITS EOR
OPWAIT STA
OPWAIT LDA
BITS+NOPS,2 EOR
NOTDUN STA
NOTDUN LDA
BITS+NOPS,2 EOR
OPSTOP STA
OPSTOP
.20.09.
... STPNXT
CARD NO. 01582
20.18--->*
03
YES, OUTPUT 'ALREADY REQUESTED' LDB OPRQCW
/01584/
04
(PTYPOT)
/01586/
05
CARRAIGE RETURN
LDB RETNCW
/01588/
06
(PTYPOT)
.22.04.
... CKWAIT
CHECK FOR ANY WAITING OPERATIONS

Fig. 14 (contd)

CHART TITLE - DIS/HSD DOI-5468-SP 1/6/76

```
        CARD NO. 01592

         20.13--->*
                  !          01
        *-------------------*
        ! MRG  NOTDUN       !
        !  NO, SET IT STA   !
        ! NOTDUN       STX  !
        ! OPIDX        CXB  !
        !              LDX  !
        !   =OPBLOK         !
        *-------------------*
                  !
                  !
          /01602/!           02
        *-------------------*
        !1!                H
        !3!   (QSTORE)     H
        !.!   STORE IN     H
        !1!  OPERATIONS    H
        !1!    QUEUE       H
        *-------------------*
                  !
                  !
          /01604/!           03
        *-------------------*
        ! LDX  OPIDX        !
        ! LDA  BITS+NOPS,2  !
        ! MRG  OPWAIT       !
        ! SET WAIT BIT STA  !
        !   OPWAIT          !
        *-------------------*
                  !                          CARD NO. 01618
                  !
        CARD NO. 01612           *------------------->*
                  !              !                    !             07
         20.11--->!              !          *--------------------*
         CKWAIT   !          04  !          ! SUB  =1            !
        *-------------------*    !          !  SET COUNTER TO    !                CARD NO. 01640
        ! LDA  OPCT         !    !          ! CHECK WAIT QUEUE   !
        *-------------------*    !          ! STA  OPCNTR        !
                  !              !          ! LDA  MASKFG        !    *------------------->*
                  !              !          ! QUEUE HOLDS INDEX  !    !                    !         12
          /01614/*      05       !          !   OF WAITING       !    !          *-------------------*
               *    *            !          !  OPERATION STA     !    !          ! LDA  OPBUSY       !
             *        *          !          !  OPBUSY            !    !          *-------------------*
           *            *  YES   !          *--------------------*    !                    !
         *   A GREATER    *------*                    !               !                    !
           *  THAN =0   *                             !               !           DOES THIS OPERATION
             *        *                      CARD NO. 01626           !           CONFLICT WITH
```

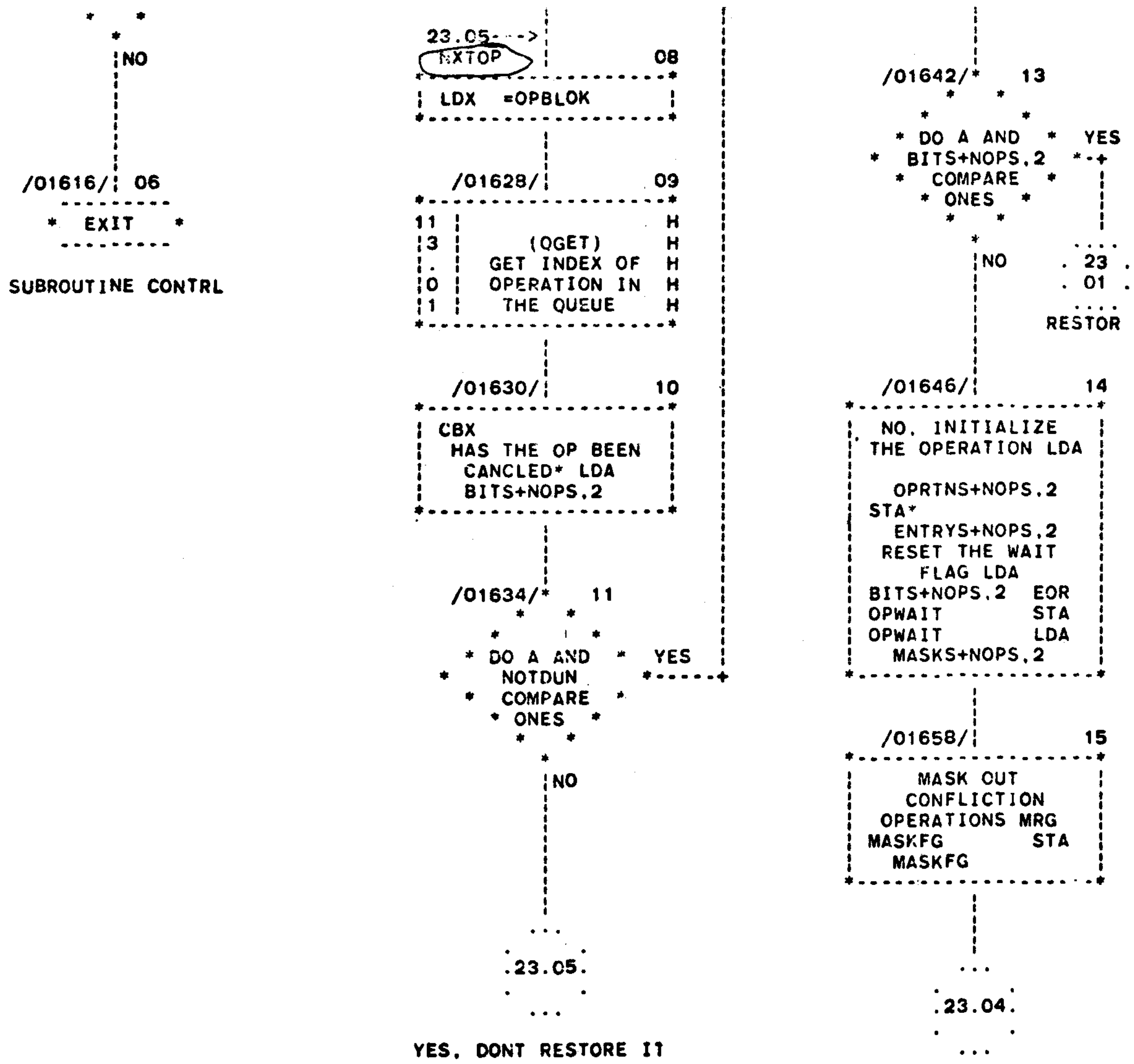

NO
/01616/ 06
EXIT
SUBROUTINE CONTRL
23.05-->
NXTOP 08
LDX =OPBLOK
/01628/ 09
11
3
.
0
1
(QGET)
GET INDEX OF
OPERATION IN
THE QUEUE
H
H
H
H
H
/01630/ 10
CBX
HAS THE OP BEEN
CANCLED* LDA
BITS+NOPS,2
/01634/* 11
DO A AND
NOTDUN
COMPARE
ONES
YES
NO
23.05
YES, DONT RESTORE IT
/01642/* 13
DO A AND
BITS+NOPS,2
COMPARE
ONES
YES
NO
23
01
RESTOR
/01646/ 14
NO, INITIALIZE
THE OPERATION LDA
OPRTNS+NOPS,2
STA*
ENTRYS+NOPS,2
RESET THE WAIT
FLAG LDA
BITS+NOPS,2 EOR
OPWAIT STA
OPWAIT LDA
MASKS+NOPS,2
/01658/ 15
MASK OUT
CONFLICTION
OPERATIONS MRG
MASKFG STA
MASKFG
23.04

Fig. 14 (contd)

CHART TITLE - DIS/HSD DOI-5468-SP 1/6/76

```
CARD NO. 01664

 22.13--->*
 RESTOR  !               01
*-------------------------*
! LDX  =OPBLOK            !
*-------------------------*
         !
   /01666/!              02
*-------------------------*
11 !                      H
!3 !    (QSTORE)          H
!. !  RESTORE THE         H
!1 !   OPERATION          H
!1 !                      H
*-------------------------*
         !
   /01668/!              03
*-------------------------*
! CBX                     !
*-------------------------*
         !
CARD NO. 01670
 22.15--->!
                         04
*-------------------------*
! LDA  MASKS+NOPS,2       !
!     MASK ANY            !
!    OPERATIONS           !
! CONFLICTING WITH        !
! MRG  OPBUSY             !
!     THIS STA            !
!   OPBUSY                !
*-------------------------*
         !
ANY MORE*
CARD NO. 01676


                 CARD NO. 01690
+------------------->*
!                    !          08
!          *---------------------*
!          ! CLA                 !
!          !     RESET THE       !
!          ! OPERATION'S INDEX   !


THIS ROUTINE STOPS AN
ONGOING OPERATION
CARD NO. 01796

 24.02*--->*
 RELES   !               14
*-------------------------*
!      START OF           !
! SUBROUTINE RELES        !
*-------------------------*
         !
   /01798/!              15
*-------------------------*
! CBA                     !
! RESET NOTDUN FLAG       !
! EOR  NOTDUN             !
! STA  NOTDUN             !
! CBA                     !
! EOR  OPSTOP             !
! STA  OPSTOP             !
!     FIND THE            !
! OPERATION:S INDEX       !
! LDX  =-NOPS             !
! CLA                     !
*-------------------------*
         !
   /01814/!              16
*-------------------------*
! STA  MASKFG             !
! LDA  OPWAIT             !
!     FIND ALL            !
!   OPERATIONS IN         !
!   PROGRESS EOR          !
! NOTDUN         CAB      !
!                         !
*-------------------------*
         !
CARD NO. 01822
+----------->!
```

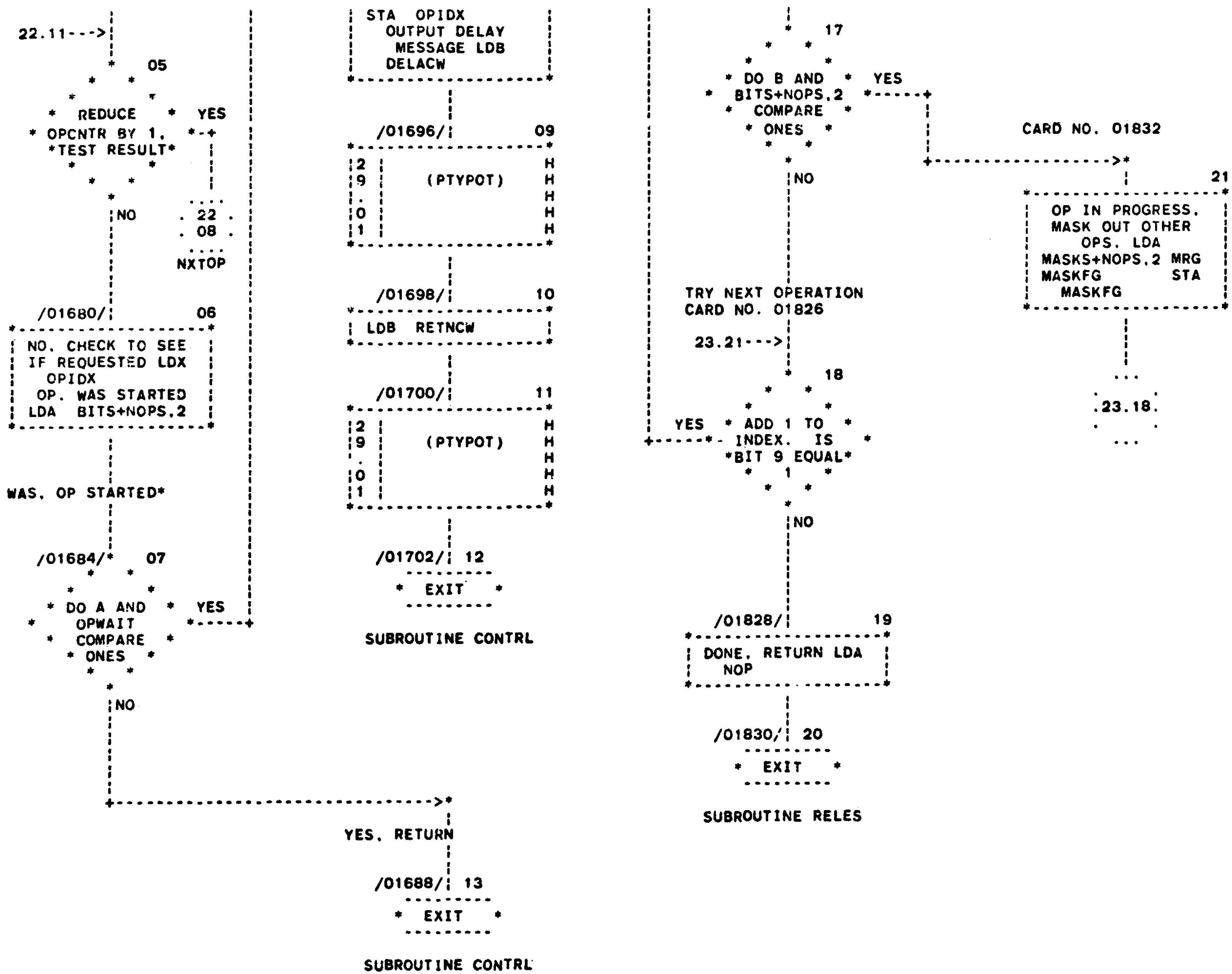

22.11--->
05
REDUCE
OPCNTR BY 1,
TEST RESULT
YES
NO
22
08
NXTOP
/01680/
06
NO. CHECK TO SEE
IF REQUESTED LDX
OPIDX
OP. WAS STARTED
LDA BITS+NOPS,2
WAS, OP STARTED*
/01684/
07
DO A AND
OPWAIT
COMPARE
ONES
YES
NO
YES, RETURN
/01688/
13
EXIT
SUBROUTINE CONTRL
STA OPIDX
OUTPUT DELAY
MESSAGE LDB
DELACW
/01696/
09
(PTYPOT)
/01698/
10
LDB RETNCW
/01700/
11
(PTYPOT)
/01702/
12
EXIT
SUBROUTINE CONTRL
17
DO B AND
BITS+NOPS,2
COMPARE
ONES
YES
NO
TRY NEXT OPERATION
CARD NO. 01826
23.21--->
18
ADD 1 TO
INDEX. IS
BIT 9 EQUAL
1
YES
NO
/01828/
19
DONE, RETURN LDA
NOP
/01830/
20
EXIT
SUBROUTINE RELES
CARD NO. 01832
21
OP IN PROGRESS,
MASK OUT OTHER
OPS. LDA
MASKS+NOPS,2 MRG
MASKFG STA
MASKFG
23.18

Fig. 14 (contd)